Universidad de Málaga
Departamento de Lenguajes y Ciencias de la Computación
Campus de Teatinos, 29071 MÁLAGA

Examen de Traductores, Intérpretes y Compiladores.
Convocatoria extraordinaria de Diciembre de 2007
3er Curso de I.T. Informática de Sistemas.

Apellidos, Nombre: _____

TEORÍA

1.- La siguiente gramática no es LALR(1) y YACC da problema cuando intenta metacompilarla:

```
sent    :       ID infijo1 PYC sent
        |       ID infijo2 PYC
        ;
infijo1:        ASIG infijo1
        |       IF
        |       ASIG
        ;
infijo2:        ASIG
        |       infijo2 ASIG
        ;
```

Se pide convertirla en otra equivalente que sí sea LALR(1).

2.- Sea la gramática

listaDecl' —> listaDecl
listaDecl —> ε | listaDecl decl PYC
decl —> ID DOSP tipo | ID COMA decl
tipo —> INT | REAL | STRUCT listaDecl END STRUCT

cuya tabla de análisis LALR(1) es la que sigue:

Tabla LALR(1)

Tabla Acción

	PYC	ID	DOSP	COMA	INT	REAL	END	STRUCT	$
0		r 2							r 2
1		d 3							Aceptar
2	d 4								
3			d 5	d 6					
4		r 3					r 3		r 3
5					d 8	d 9		d 10	
6		d 3							
7	r 4								
8	r 6								
9	r 7								
10		r 2					r 2		
11	r 5								
12		d 3					d 13		
13								d 14	
14	r 8								

Tabla Ir A

	listaDecl	decl	tipo
0	1		
1		2	
2			
3			
4			
5			7
6		11	
7			
8			
9			
10	12		
11			
12		2	
13			
14			

Se pide aceptar o rechazar la cadena:

ID DOSP INT PYC ID COMA ID DOSP STRUCT END STRUCT PYC $

indicando en cada paso la configuración alfa-beta (pila-cadena por reconocer).

3.- Construir **un único** diagrama de sintaxis que reconozca el mismo lenguaje que la gramática anterior.

4.- Explicar brevemente los métodos de recuperación de errores sintácticos que conozca.

Open Course Ware
Gálvez Rojas, S.
(2011). Traductores, Compiladores e Intérpretes
OCW- Universidad de Málaga http://ocw.uma.es
Bajo licencia Creative Commons Attribution-Non-Comercial-ShareAlike
CC BY NC SA

Solución:

```
%token ID PYC ASIG IF
%%
sent    :       ID infijo1
        ;
infijo1 :       infijo2 IF PYC ID infijo1
        |       infijo2 PYC ID infijo1
        |       infijo2 PYC
        ;
infijo2 :       ASIG
        |       infijo2 ASIG
        ;
```

Otra solución:

```
%token ID PYC ASIG IF
%%
sent    :       ID infijo2 ifOpcional sent
        |       ID infijo2 PYC
        ;
ifOpcional : IF PYC
        | PYC
        ;
infijo2 :       ASIG
        |       infijo2 ASIG
        ;
```


Bajo licencia Creative Commons Attribution-Non-Comercial-ShareAlike

Universidad de Málaga
Departamento de Lenguajes y Ciencias de la Computación
Campus de Teatinos, 29071 MÁLAGA

Examen de Traductores, Intérpretes y Compiladores.
Convocatoria extraordinaria de Diciembre de 2007
3er Curso de I.T. Informática de Sistemas.

Apellidos, Nombre: _____

Calificación: _____

Se pretende ampliar ligeramente la gramática propuesta en clase para generar código de tercetos. Concretamente se desean introducir las siguientes ampliaciones:

a) Una expresión puede adoptar la forma de las expresiones condicionales propias de C y de Java representadas mediante la sintaxis:

expr : '(' cond ')' '?' expr_1 ':' expr_2

de manera que la **expr** al completo tomará el valor de expr_1 si la **cond** es verdadera y el de expr_2 si la **cond** es falsa.

b) Las condiciones se ven ampliadas de tres formas distintas:

b.1) Se permite el uso del operador IMPLIES (implicación) cuya funcionalidad obedece a la tabla de verdad de la figura[1]:

cond : cond_1 IMPLIES cond_2

b.2) Se permite el uso del operador XOR (OR exclusivo) cuya funcionalidad obedece a la tabla de verdad de la figura:

cond : cond_1 XOR cond_2

a	b	a IMPLIES b	a XOR b
F	F	V	F
F	V	V	V
V	F	F	V
V	V	V	F

b.3) Aparece un nuevo tipo de condición *simple* que permite comprobar si una determinada expresión coincide o no con alguno de los valores dados por una serie de expresiones:

cond: expr_1 IN '(' listaExpr ')'
listaExpr : expr
| listaExpr ',' expr

Si expr_1 coincide con alguno de los valores de la lista, entonces se considera la **cond** como verdadera, y falsa en caso contrario.

A continuación se muestra el código que puede generarse ante determinadas sentencias de

[1] Recuérdese que, según el álgebra de Boole, se verifica que: $a \rightarrow b \equiv \neg a \vee b$

Open Course Ware
Gálvez Rojas, S.
(2011). Traductores, Compiladores e Intérpretes
OCW- Universidad de Málaga http://ocw.uma.es
Bajo licencia Creative Commons Attribution-Non-Comercial-ShareAlike
BY NC SA

entrada:

Código Fuente	Código generado	

IF a+b IN (14, 32, 56+2) THEN
c:=1;
FIN IF;
	tmp1 = a + b;
	tmp2 = 14;
	if tmp1 = tmp2 goto etq1
	tmp3 = 32;
	if tmp1 = tmp3 goto etq1
	tmp4 = 56;
	tmp5 = 2;
	tmp6 = tmp4 + tmp5;
		if tmp1 = tmp6 goto etq1
		goto etq2
		label etq1
		tmp7 = 1;
		c = tmp7
		goto etq3
		label etq2
		label etq3

IF (a>b) XOR (a>c) THEN
c:=10;
FIN IF;
	if a > b goto etq4
	goto etq5
	label etq4
	tmp8 = 1
	goto etq6
	label etq5
	tmp8 = 0
	label etq6
	if a > c goto etq7
	goto etq8
	label etq7
	if tmp8=1 goto etq10
	goto etq9
		label etq8
		if tmp8=0 goto etq10
		goto etq9
		label etq9
		tmp9 = 10;
		c = tmp9
		goto etq11
		label etq10
		label etq11

IF (a>b) IMPLIES (a>c) THEN
c:=100;
FIN IF;
	if a > b goto etq12
	goto etq13
	label etq12
	if a > c goto etq14
	goto etq15
	label etq13
	goto etq14
	label etq14
	tmp10 = 100;
	c = tmp10
	goto etq16
	label etq15
	label etq16	

a := (b>c)? v : f ;	if b > c goto etq17
	goto etq18
	label etq17
	tmp11 = v;
	goto etq19
	label etq18
	tmp11 = f;
	label etq19
	a = tmp11	

Bajo licencia Creative Commons Attribution-Non-Comercial-ShareAlike

Se pide:

- Indicar los atributos necesarios a los símbolos de la gramática. No es necesario modificar el **%union** visto en clase ya que es posible apañárselas con lo ya existente en él.
- Rellenar el esqueleto de YACC proporcionado (la parte Lex modificada no es necesario proporcionarla debido a su simplicidad).

Atributos:

- No terminal **interrogacion**: _____
- Terminal **XOR**: _____
- No terminal **inicio_in**: _____

Esqueleto:

```
expr:   /* Lo ya visto en clase... */
        |       '(' cond ')' interrogacion       {
                                                 }
                                                 expr ':'                {
                                                 }
                                                 expr %prec IF_ESPECIAL {
                                                 // El token IF_ESPECIAL puede obviarse. Es sólo para que no aparezcan
                                                 conflictos shift/reduce
                                                 }
        ;
interrogacion  :        '?'     {
                                                 }
                                                 ;

cond:   /* Lo ya visto en clase... */
        |       cond IMPLIES
                                                 {
                                                 }
                cond            {
                                                 }
```


Bajo licencia Creative Commons Attribution-Non-Comercial-ShareAlike

```
        |       cond XOR
                        {

                        }
                cond    {

                        }
        |       cond_in
        ;
cond_in     :       inicio_in ')'     {

                        }
                ;
inicio_in   :       expr IN '(' expr          {

                        }
                    |       inicio_in ',' expr        {

                        }
                    ;
```


Bajo licencia Creative Commons Attribution-Non-Comercial-ShareAlike